

| 1. AGENCY USE ONLY (Leave blank) | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | March 1993 | Professional Paper |

4. TITLE AND SUBTITLE

TRAINING NEURAL NETWORKS WITH WEIGHT CONSTRAINTS

5. FUNDING NUMBERS

In House Funding

6. AUTHOR(S)

J. R. McDonnell

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Command, Control and Ocean Surveillance Center (NCCOSC)
RDT&E Division
San Diego, CA 92152-5001

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Naval Command, Control and Ocean Surveillance Center (NCCOSC)
RDT&E Division
San Diego, CA 92152-5001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER



11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

Hardware implementation of artificial neural networks imposes a variety of constraints. Finite weight magnitudes exist in both digital and analog devices. Additional limitations are encountered due to the imprecise nature of hardware components. These constraints can be overcome with a stochastic global optimization strategy which effectively searches the range of the weight space and is robust to quantization and modeling errors. Evolutionary programming is proposed as a solution to training networks with these constraints. This work investigates the use of evolutionary programming in optimizing a network with weight constraints. Comparisons are made to the backpropogation training algorithm for networks with both unconstrained and hard-limited weight magnitudes.

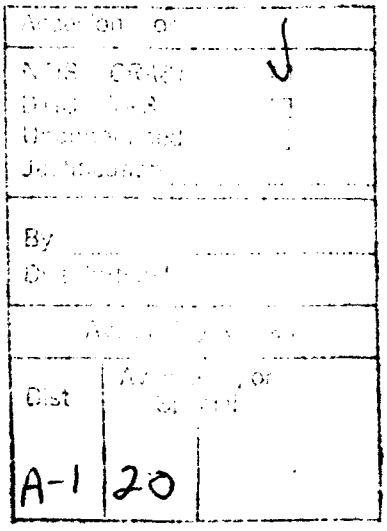


Published in *Proceedings*, First Annual Conference on Evolutionary Programming, pp. 111-119.

| 14. SUBJECT TERMS | | 15. NUMBER OF PAGES |
|---|---|---|
| neural networks | analog | |
| digital | stochastic | 16. PRICE CODE |

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| UNCLASSIFIED | UNCLASSIFIED | UNCLASSIFIED | SAME AS REPORT |

| 21a. NAME OF RESPONSIBLE INDIVIDUAL | 21b. TELEPHONE (Include Area Code) | 21c. OFFICE SYMBOL |
|---|---|---|
| J. R. McDonnell | (619) 553-5762 | Code 731 |



# Training Neural Networks with Weight Constraints

*John R. McDonnell*
*NCCOSC, RDT&E Division*
*San Diego, CA 92152*

## ABSTRACT

*Hardware implementation of artificial neural networks imposes a variety of constraints. Finite weight magnitudes exist in both digital and analog devices. Additional limitations are encountered due to the imprecise nature of hardware components. These constraints can be overcome with a stochastic global optimization strategy which effectively searches the range of the weight space and is robust to quantization and modeling errors. Evolutionary programming is proposed as a solution to training networks with these constraints. This work investigates the use of evolutionary programming in optimizing a network with weight constraints. Comparisons are made to the backpropagation training algorithm for networks with both unconstrained and hard-limited weight magnitudes.*

## INTRODUCTION

Hardware implementation of neural networks poses many challenges. The magnitude constraint imposed by implementing neural network weights in electronic devices artificially constrains the range of the weight space over which the search takes place. Further, standard off-line training approaches may produce poor results due to the imprecise nature of analog devices. This work investigates the use of a stochastic multi-agent search strategy for optimizing feed-forward, fully connected, neural networks. An evolutionary programming (EP) search strategy is proposed as a weight adaptation scheme which addresses these constraints.

A variety of adaptation difficulties can be associated with hardware implementation of neural networks. Hardware issues which greatly affect learning include finite magnitude weights, quantization error and imprecise knowledge of the activation function. Weight and node perturbation strategies have been suggested [1,2] as an alternative to using error back-propagation for training analog implementations. These strategies provide an empirical gradient descent approach given imprecise *a priori* knowledge of the analog activation function In studying quantization error, Xie and Jabri [3,4] have applied a statistical model to determine the effects of quantization error in digital implementations.

A modified weight perturbation strategy has been developed [4] as an empirical gradient training technique for digital implementations. As pointed out in [4], quantized weight values represented by a low number of bits can lead to local minima which are flat plateaus. Thus, a gradient type search may yield poor training performance for a digital implementation. To overcome this limitation, Xie and Jabri incorporate a stochastic mechanism which results in a "Combined Search" (CS) algorithm. The CS algorithm alternates between the modified weight perturbation technique and a partial random search (PRS). The PRS is applied to only one dimension of the weight space wherein a

single weight is randomly chosen from a uniform distribution over a predetermined range. The randomly chosen weight is kept if a lower network error is achieved.

The ability to avoid local minima is but one advantage in using a stochastic search technique. Other advantages include the applicability of stochastic training to any type of architecture regardless of connectivity, activation function continuity, and *a priori* knowledge of the device characteristics.

Neural network training using stochastic methods is not uncommon. Simulated annealing has been applied to both feedforward [5] and Hopfield [6,7] networks. Convergence aspects of the simulated annealing cooling schedule have been addressed by Geman and Geman [8]. Genetic algorithms [9] offer yet another stochastic method for training neural networks. A probabilistic backpropagation technique has been developed for hard-limited activation nodes [10]. Fogel *et al.* [11,12] have taken an evolutionary programming approach to stochastically evolve network weights.

This work takes advantage of the evolutionary programming paradigm to evolve network weights which are subject to weight magnitude constraints. Problems associated with quantization errors and transfer characteristics of the device can also be addressed with this type of training. A comparison is made with backpropagation training for both unconstrained and constrained weights. For situations where the device transfer characteristics are known with a high degree of certainty, a modified backpropagation technique is developed and presented in the next section.

## MODIFIED BACK PROPAGATION

If weight magnitudes are hard-limited within the backpropagation training algorithm, optimization will occur along the projections of the saturated weight space dimensions. In essence, optimization will take place in the image of the unconstrained weight space. To avoid being limited to the weight space between $m$ saturated weights and an $(n\text{-}m)$ dimensional ridge (if one exists), a modified backpropagation (MBP) training algorithm is developed. Wasserman [13] has proposed function mapped weights as a means to reduce neuron saturation. If function mapped weights are employed, then the function which maps the training weights to the implementation weights must be incorporated into the training algorithm. Figure 1 shows a perceptron which implements function mapped weights as opposed to the weight values directly. The weight values can be determined via offline training and implemented by the weight mapping function.

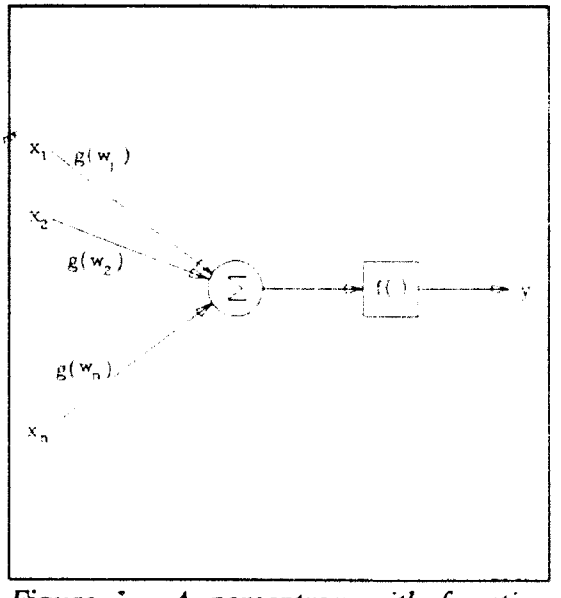



*Figure 1. A perceptron with function mapped weights g(w).*

The training algorithm can be derived using the same approach developed for backpropagation [14]. The primary difference is that the weight mapping function $g$ must be incorporated in the training algorithm. This is done by defining the gradient as

$$\nabla = \frac{\partial E}{\partial w} = \frac{\partial E}{\partial g(w)} \frac{\partial g(w)}{\partial w}$$

where $E$ is the network sum squared error and $g$:$\mathbf{R}\rightarrow\mathbf{R}$. Using the generalized delta rule for feedforward networks with backpropagation of error as outlined in [14], the weight adaptation equations can be written as

*MBP output layer*

$$\Delta w_{kj} = \eta f_k'(net_k)(t_k - o_k) g_{kj}'(w_{kj}) o_j$$

*MBP hidden layer*

$$\Delta w_{ji} = \eta f_j'(net_j)\left[\sum_k \delta_k\, g(w_{kj})\right] g_{ji}'(w_{ji}) o_i$$

where

$$g_{ji}' = \frac{\partial g(w_{ji})}{\partial w_{ji}}$$

the remaining terms are defined in [14].

It is easily verified that the above learning law reduces to the standard generalized delta learning rule if the weight mapping function is replaced with the weights themselves (*i.e.*, $g(w) = w$, $g'(w) = 1$)

*BP output layer*

$$\Delta w_{kj} = \eta f_k'(net_k)(t_k - o_k) o_j$$

*BP hidden layer*

$$\Delta w_{ji} = \eta f_j'(net_j)\left[\sum_k \delta_k w_{kj}\right] o_i$$

Similar to the activation function $f$, the weight mapping function $g$ must also be continuously differentiable. Additionally, $g$ should also be bipolar. Since a biased sigmoid function satisfies this requirement, it is a likely (though not necessarily optimal) candidate to be used for function mapped weights

If the activation $f$ is a unipolar sigmoid function and the weight mapping $g$ is a bipolar sigmoid function of magnitude $K$

$$g(w) = 2K[(1+e^{-w})^{-1} - \tfrac{1}{2}]$$

then the weights are modified according to

*Output layer*

$$\Delta w_{kj} = \eta\, o_k(1-o_k)(t_k - o_k)\, o_j * \frac{1}{2K}\left[g(w_{kj}) + K\right]\left[K - g(w_{kj})\right]$$

*Hidden layer*

$$\Delta w_{ji} = \eta\, o_j(1-o_j)\left[\sum_k o_k(1-o_k)(t_k - o_k) g(w_{kj})\right] * \frac{1}{2K}\left[g(w_{ji}) + K\right]\left[K - g(w_{ji})\right]$$

The MBP algorithm effectively constrains the weight set magnitudes while maintaining the integrity of the gradient descent search. The ability to implement such a training technique is predicated on modeling the transfer characteristics of the activation and weight mapping functions. If accurate device modeling is not feasible, then alternative training methods should be employed.

## EVOLUTIONARY PROGRAMMING

Evolutionary programming (EP) is a neo-Darwinian search paradigm suggested by Fogel *et al.* [15]. The application of EP as a neural net training technique has been investigated by Fogel *et al.* [11,12]. EP is used in these investigations since it addresses the difficulties imposed with hardware implementation: finite weight magnitudes and imprecise knowledge of the device characteristics and parameters. The main benefit of using an EP approach is that it provides an efficient global search of the constrained weight space. While Stinchcombe and White [16] have shown that multilayer feedforward networks with bounded weights and various classes of activation functions can learn arbitrary mappings, they did not specify a particular learning technique.

Other global stochastic search techniques were considered. Genetic algorithms are just a subset of the EP paradigm. The EP paradigm does not restrict representation or type of mutation process. Although GA does use dramatically different mutations (*e.g.*, crossover mutation), it is easily encompassed by the EP method. The multi-agent search of EP is preferred to the single-agent approach of simulated annealing with its various cooling schedules. However, simulated annealing has been successfully implemented at the device level as discussed in [17].

The objective function of the EP training approach is the same as that used in backpropagation: minimize the error function $E = \frac{1}{2}\Sigma_p\Sigma_k (t_k - o_k)^2$ where $p$ is the pattern training set and $k$ is the number of output nodes. A common metric is the mean of $E$ over the number of patterns which will be referred to later as the MSE of the network. This investigation assumes fully connected feedforward networks. However it should be noted that EP could be used to simultaneously evolve network structure, connectivity and connection strengths.

An EP algorithm for training a neural network can be described by the following steps

1. *Form an initial population of size N.*
2. *Assign a fitness score to each element (network) of the population.*
3. *Reorder the population based on the number of wins generated from a stochastic competition process.*
4. *Generate offspring of the best N networks.*
5. *Loop to step 2.*

For this application, the initial population was instantiated with weights $\epsilon$ *U[-0.5, 0.5]*. Each member (network) in the population is represented by a four dimensional weight array

$$\Phi_i = w[parent][layer][from\_node][to\_node]$$

Next, a cost is assigned to each network in the population. As previously mentioned, the cost or objective function used in this study is the same as that used for backpropagation. The $N$ members of the population generate offspring according to the perturbation

$$W_o = W_p + \delta W_p$$

where typically $\delta W_p \epsilon N(0, S_F \cdot E)$ with a scaling coefficient $S_F$. The scaling factor is a probabilistic analog to the stepsize in gradient descent methods. Figure 2 shows the effect of different scaling factors in training a *2-2-1* neural net for the *XOR* mapping.

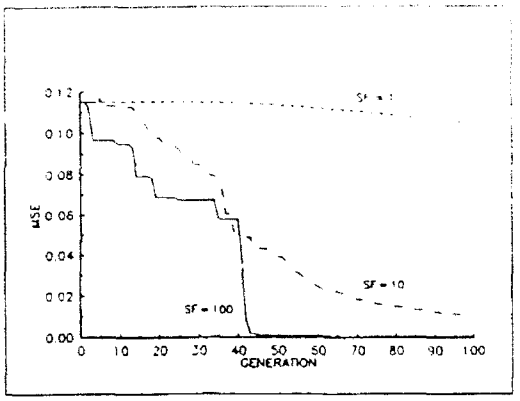



*Figure 2. The effect of various scaling factors on net training for XOR function.*

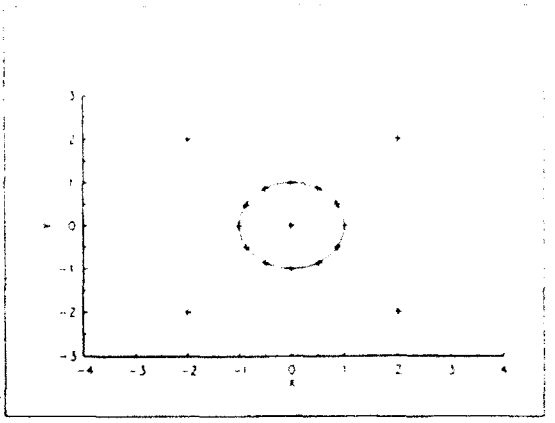


*Figure 3. Training points for unit circle mapping problem.*

A pairwise competition is then held where individual elements compete against randomly chosen members of the population. If network $\Phi_j$ is randomly selected to compete against network $\Phi_i$, then a win is awarded to $\Phi_i$ if $E_i < E_j$. The $N$ networks with the most wins are generate offspring and the process is repeated.

## RESULTS

### Hard Limited Weights

An arbitrary mapping problem of determining whether an $<x,y>$ point is contained within the unit circle has been chosen for evaluating the effectiveness of hard-limited backpropagation versus the EP training approach. The training set consists of 17 data points with a target value of 0.98 if the point is contained within the unit circle and 0.02 if the point is outside the unit circle. Figure 3 shows the distribution of the points selected. A *17-8-1* network was used in all experiments. A population size of $N = 50$ parents with 50 offspring was used for the EP runs. The effect of hard limiting the weights in backpropagation is shown in Figure 4. It can be seen that backpropagation training ($\eta = 0.5$, $\alpha = 0.1$) can easily form the necessary hyperplane decision surfaces to separate the convex unit circle if no weight constraints are imposed. This run resulted in 61% of the network weights greater than a magnitude of 5. The problem becomes more difficult if the maximum weight magnitudes are hard-limited at 5. The stepsize and momentum coefficients were reduced to 0.1 and 0.05, respectively, to prevent oscillation. The constrained network reached a final MSE of 0.0297 with 36% of the weights within 1% of saturation.

The same experiment was repeated using EP. Figure 5 shows the MSE for the best member (network) in the population at each generation for both constrained and unconstrained weight magnitudes. Scaling

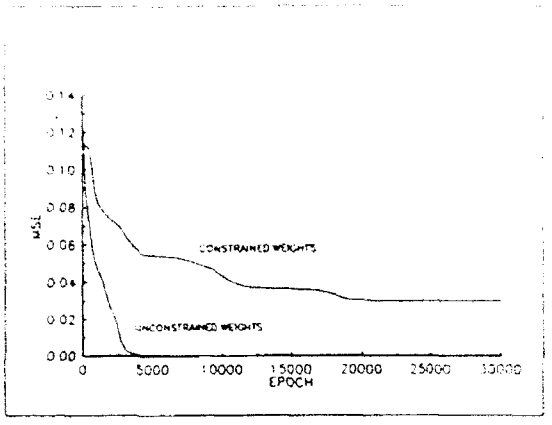



***Figure 4. Backpropagation training for constrained and unconstrained networks.***

factors of 100 and 10 were used for the unconstrained and constrained networks, respectively. The unconstrained trial resulted in 91% of the weights greater that a magnitude of 5. The constrained trial reached a final MSE of 0.0208 with 36% of the weights within 1% of maximum magnitude.

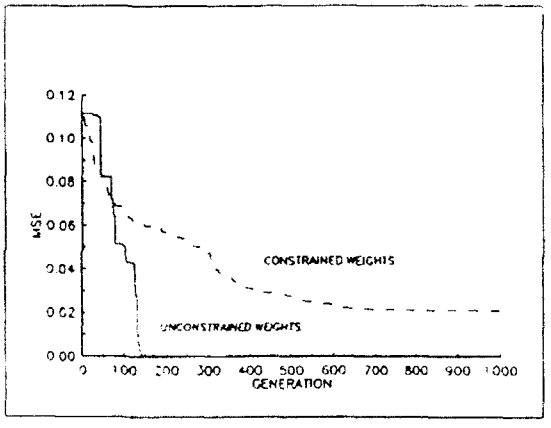



***Figure 5. EP training results for constrained and unconstrained networks.***

### Hard Limited Neuron Activations

Gradient descent algorithms are not amenable to training multilayer perceptrons with hard limited activation functions. Bartlett and Downs [10] have developed a probabilistic backpropagation analog for training multilayer perceptrons with hard-limited activation functions. Since the EP training technique is applicable to any architecture with any activation function, it was applied to the unit circle mapping problem with hard-limited activation functions.

This set of experiments incorporated a binary activation function

$$f(x)=\begin{cases}1 \text{ if } x\geq 0\\ 0 \text{ if } x<0\end{cases}$$

at each of the neuron outputs. Two trials were run to compare the effects of hard-limiting the weight magnitudes. The results of this experiment are shown in Figure 6.

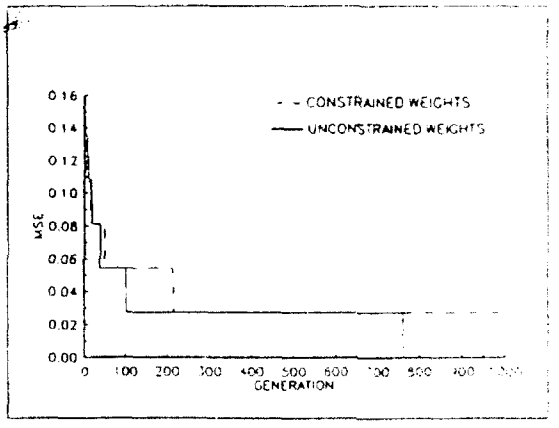



***Figure 6. EP training with hard limited activation functions. Scale factor = 100.***

Both experiments used a scaling factor of 100. The unconstrained weight network was able to reach zero MSE within 1000 generations. Out of the initial 50 parents, the best parent was incorrect on just 5 patterns. MSE minimization takes place in discrete steps due to the hard-limit activation function. The unconstrained network had 94% of its weight values greater than 5. Applying a weight magnitude bound at 5 resulted in a MSE of 0.0271 after 1000 generations (50,000 function evaluations). None of the constrained network weights were within 1% of the saturation magnitude.

## Hybrid Learning

Given that EP provides a global search with asymptotic convergence properties, can further error reduction be achieved using gradient descent methods? This section investigates applying backpropagation to networks with bounded weight magnitudes after an arbitrary number of global (EP) iterations have been completed. Of course, this is only applicable if the activation function is accurately modeled. If the activation function cannot be accurately modeled, then a hybrid EP/direction set technique similar to that given by Waagen *et al.* [18] can be used. Another approach would be to incorporate the local methods developed in [1, 2 and 4] in conjunction with EP.

Figure 7 shows the training results for the best member in the population over 3000 generations (150000 pattern set function evaluations) for the unit circle mapping problem. At the iteration 3000 the best member in the population has a MSE of 0.0206. A backpropagation search with hard-limited weight constraints was applied to this network. This refinement yielded approximately a 1.6% decrease in the MSE after 10000 epochs as shown in Figure 8. The results shown in Figure 8 are for $\eta=0.01$ and $\alpha=0.002$. If too large a stepsize is chosen the initial MSE can actually increase before refinement takes place.

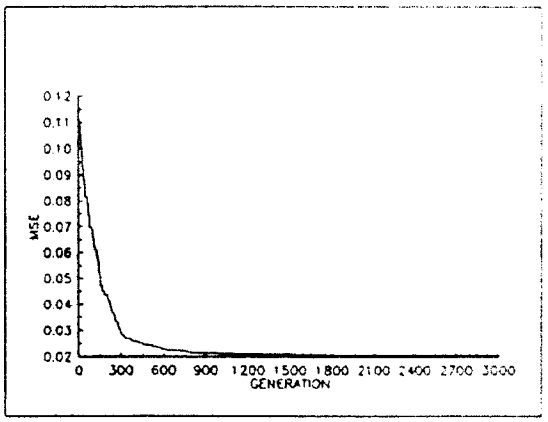



***Figure 7. EP training results for constrained network.***

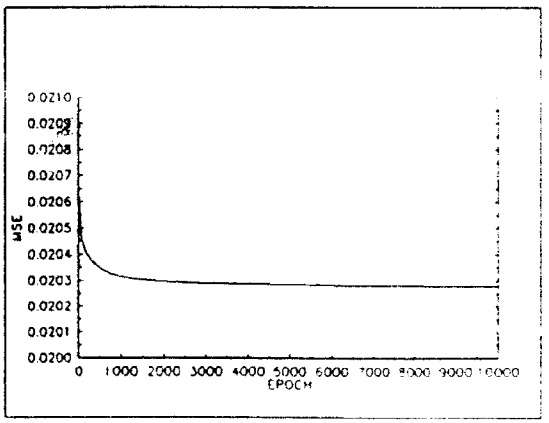



***Figure 8. Constrained backpropagation refinement of 'best' EP weight set from figure 7.***

## DISCUSSION

This study shows that EP can yield results as good as or better than a backpropagation training approach for both bounded and unbounded weights. EP is well suited for training neural networks subject to the additional constraints and inaccuracies encountered in hardware implementations whereas backpropagation might not be appropriate. If the activation functions are well known, then a modified backpropagation method can be used which maintains the integrity of the gradient descent search. It is not known at this time if this technique increases the likelihood of local minima or has other ramifications. Hybrid learning can be applied to networks regardless of how well the activation function is known. The example given in this work shows additional optimization occurs after the global search has been arbitrarily stopped. A reasonable approach would be to incorporate a global and local search in parallel. The inability of traditional methods to train a multilayer perceptron with hard-limited activations can be overcome using a stochastic method such as EP.

Minsky and Papert [19] state that stochastic methods will not "... *work very well on large-scale nets, except in the case of problems that turn out to be of low order in some appropriate sense*". Nevertheless, EP provides a systematic way of training a network subject to a variety of constraints and lack of *a priori* knowledge where other techniques fall short. However, the memory requirements associated with stochastic techniques which optimize a "population" of networks may be substantial. Minsky and Pappert go on to say that "*The power of the brain ... comes from the evolution (in both the individual sense and the Darwinian sense) of a variety of ways to develop new mechanisms and to adapt older ones to perform new functions.*" This idea of interacting mechanisms can be easily incorporated into the paradigm of simulated evolution [15].

## REFERENCES


1. D. Andes, B. Widrow, M. Lehr, and E. Wan, "MRIII: A Robust Algorithm for Training Neural Networks", Int. Joint Conf. on Neural Networks, Washington, D. C., 1990.

2. M. Jabri and B. Flower, "Weight Perturbation: An Optimal Architecture and Learning Technique for Analog VLSI Feedforward and Recurrent Multilayer Networks", IEEE Trans. on Neural Networks, Vol. 3., No. 1, Jan. 1992.

3. Y. Xie and M. Jabri, "Analysis of the Effects of Quantization in Multilayer Neural Networks Using a Statistical Model", IEEE Trans. on Neural Networks, Vol. 3, No. 2, March 1992.

4. Y. Xie and M. Jabri, "Training Algorithms for Limited Precision Feedforward Neural Networks", SEDAL Technical Report No. 1991-8-3, 1991.

5. S.P. Day and D.S. Camporese, "A Stochastic Training Technique for Feed-Forward Neural Networks", Int. Joint Conf. on Neural Networks, San Diego, 1990.

6. G.E. Hinton and T.J. Sejnowski, "Learning and Relearning in Boltzman Machines", in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*, D.E. Rumelhart and J.L. McClelland (Eds.), MIT Press, 1986.

7. E. Aarts and J. Korst, *Simulated Annealing and Boltzman Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*, John Wiley & Sons, 1989.

8. S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images", IEEE Proc. Pattern Analysis and Machine Intelligence, Vol. 6, 1984.

9. D.H. Ackley, "Stochastic Iterated Genetic Hill-Climbing", Ph. D. dissertation, Computer Science Dept., CMU-CS-87-107, Carnegie Mellon

University, Pittsburgh, PA, 1987.

10. P.L. Bartlett and T. Downs, "Using Random Weights to Train Multilayer Networks of Hard-Limiting Units", IEEE Trans. on Neural Networks, Vol. 3, No. 2, March, 1992.

11. D.B. Fogel, L.J. Fogel, and V.W. Porto, "Evolving Neural Networks", Biological Cybernetics, 63, 1990.

12. D.B. Fogel and A.V. Sebald, "Training Neural Networks through Evolutionary Adaptation", AMSE Int. Conf. on Neural Networks: Methodologies and Applications, San Diego, 1991.

13. P.D. Wasserman, *Neural Computing: Theory and Practice*, Von Nostrand Reinhold, 1989.

14. D.E. Rumelhart, G. E. Hinton and R.J. Williams, " Learning Internal Representations by Error Propagation" in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*, D.E. Rumelhart and J.L. McClelland (Eds.), MIT Press, 1986.

15. L.J. Fogel, A.J. Owens and M.J. Walsh, *Artificial Intelligence through Simulated Evolution*, John Wiley & Sons, 1966.

16. M. Stinchcombe and H. White, "Approximating and Learning Unknown Mappings using Multilayer Feedforward Networks with Bounded Weights", Int. Joint Conf. on Neural Networks, San Diego, 1990.

17. B.W. Lee and B.J. Shen, "Analysis and Design of Analog VLSI Neural Networks", in *Neural Networks for Signal Processing*, B. Kosko (Ed.), Prentice-Hall, 1992.

18. D. Waagen, P. Diercks, and J. McDonnell, "The Stochastic Direction Set Algorithm: A Hybrid Technique for Finding Function Extrema", to be published in the First Annual Conf. on Evolutionary Programming, San Diego, 1992.

19. M.L. Minsky and S.A. Papert, *Perceptrons*, Expanded Edition, MIT Press, 1988.